Distributed Tasks Manager Application

Main requirements specification, phase A, 2014-01-18

Characterization

The Distributed Tasks Manager Application (DTM) application is common purpose service class software. The main functional goal is to provide possibilities for users to manage remote tasks execution and computational resources in the HCE DRCE cluster execution environment.

General features:

- Tasks management.
- Resources management.
- Execution environment management.

Conditions

Standalone boxed application that works as a client-side top user level of HCE's DRCE cluster server.

Usage Domain

Applied by developers and system integrators in target projects as server-side software. Most common purposes:

- Scheduling tasks to be executed at particular time and/or execution environment load optimization.
- Distributed asynchronous parallel tasks execution and data processing.
- Handling long and heavy job tasks like file format conversions (video, images, documents, etc).
- Offloading heavy backend jobs.
- Support any kind of executable that accepted by hosts platform.
- Load-balancing data processing algorithms.
- Failover data processing solutions.
- Distributed data close to code solutions.

Technology

Single process, multithreaded, Python language application implementation, integrated with HCE DRCE cluster server by HCE Python API bindings.

Object model

Threaded control objects, MOM-based networking, ZMQ tcp and inproc sockets, messages communications, messages queues.

Functionality

Process client requests, tasks planning and scheduling, tasks management in DRCE cluster.

Philosophy

Asynchronous multithreaded mutexless messages handling, parallel computations, asynchronous distributed remote tasks processing.

Components and objects

Task

Task it is a regular executable **artifact** that can be supported by **data nodes** of a cluster. Task is an atomic **execution unit** that can be managed and controlled. Task identified by the DRCE API specification as request with specification of command to execute, data that can be provided with task request for processing (or got via any another task-specific API) as well as the way to retrieve processing results. Tasks can be selected and managed by criterions like: group Id, user Id, creation date, scheduled date, terminated date, termination status, estimated execution time, real execution time, and fetch results counter. Tasks can be **synchronous** and **asynchronous**.

Synchronous – executed by execution environment in sequential one-by-one mode. This mode supposes that task completed in one step after request and results returned immediately. No resource planning and scheduling is involved and data flow inside cluster competes with task's computations for hardware resources like CPU, RAM and disk I/O. These tasks are typically short, a request encapsulates command (possible with program code or binary executable) and small size input data. Responses are contains processing results that possible can be reduced inside execution environment. The request can be timed out. In case of timeout is too small, results can be lost. Most effective cluster resources utilization can be reached with usage small request messages size and multi-client requester model. Possible direct reflection of external, for example web requests, to task execution requests to get scalable architecture for high loaded web-sites.

Asynchronous – executed by execution environment in scheduled lazy mode. This mode supposes that computational environment using **execution environment** resources load level controlling and resources usage **planning strategies** as well as **tasks scheduling**. Task that was set by user is not started inside execution environment right after it was created. The computational environment **tasks manager** checks the time that was requested by user as best for user's application business logic (possible nearest in time or ASAP, or in range of dates interval). To make the algorithm of prediction of task timing more effective, user can to specify estimated execution time of task and **POSIX threads** number that task potentially can to utilize. If schedule has a place (at least – the number of timeslots that equal or greater than estimated execution time) for a task it will be set and take a free place in size of predicted resources (execution time, number of POSIX threads, amount of RAM, amount of disk space and so on). If no place for the tasks, the tasks manager can to return the error and finish user request on task creation or to try to choose another time slot(s) according with strategy that was defined by create task request parameters for this kind of situation (for example ASAP for time, best for free resources (CPU, for example) and so on). After task was scheduled the task manager using clocking, time slots and

information about the current execution environment state (execution environment average load level and another resources property values) starts tasks that are set for a correspondent time slots. Tasks scheduler can to change position of the task or to skip and cancel it if some wrong state of execution environment detected on the moment of the time slot reaches now state. So, depends on strategy, scheduler can act as real-time or continues postponed. Another key part of system that takes a place in a tasks execution process – it is a **resource manager**. Execution environment based on DRCE API configured to use hardware resources like CPU, RAM, DISK and so on. These resources located on each data host of the HCE cluster and monitored periodically to compute average usage level per each resource type. The current average values of resources state indicators are actual parameters that used by scheduler during planning and processing of time slots of schedule.

Tasks chains

In asynchronous mode tasks can be **single unit** or united in to the **tasks chains**. Chains are planned and represented in the schedule as one single task. Scheduler executes chains the same way as regular separated tasks, but with execution environment limitations. For example, all chains are executed on the same HCE node (and on the same physical host, as well). The stdout from previous task became stdin for next. Behavior of the node executor depends on strategy defined for each chain unit. For example, if some chain fault executor can to abort whole task or to continue execution of next chain. The results stored the same way for each chain separately, but can to be fetched in one get results request.

Tasks sequences

Both synchronous and asynchronous modes tasks can be set in schedule as **independent** and **dependent**. **Independent** supposes that task is executed according with the strategy at the specific time and execution environment load level. Dependent acts as a sequential after another task started, worked defined time, terminated or finished. So, schedules can be created not only using the time marks and execution environment load level conditions, but tasks sequence condition.

Tasks manager

This is object that receives tasks as atomic units from user client, store task's data in the **tasks data storage**, set task to the **tasks scheduler** object and receives information about the task state from **tasks state update service**, execution environment service, and **tasks executor**. Tasks manager implements main algorithms that used to handle and manage of tasks state, set tasks to scheduler, cancel tasks, reschedule tasks, manage tasks sequences and so on.

Resource

Resource – it is a hardware or software unit of possibility or feature.

Typical **hardware resources** are CPU cores number, CPU load, RAM size, DISK size, I/O wait level, Network bandwidth level, Network connections number and so on.

The **software resources** can be represented as libraries calls, libraries instances number, modules loaded, services requests number, processes number, objects instances number and so on.

For HCE cluster running in **load balancing mode** DRCE manager uses resources of all nodes in **competitor mode**. This mode supposes average spreading of resources on all tasks' processes proportional way. But practically, the possibility to spread resources depends on OS state, hardware architecture and tasks specific conditions like **artifact execution mode** and **artifact type**.

Resource manager

It is an object that holds resources list and manage resources properties information, process messages from **resource state monitor** object and from the scheduler. Resource manager implements algorithms of resources operation and state controlling.

Artifact

Artifact or **task's artifact** – it is unit that is used to do some algorithmic job on a target host system and takes resources. Artifact spawned by **tasks executor** of **HCE node** that is a part of DRCE subsystem. Artifacts can be classified on types. One of classification based on **execution way**. It can be machine binary code in form of compiled native CPU instructions for concrete hardware platform, or p-code of some virtual machine like JVM, or human readable source code for just-in-time or interpreter real-time execution including the shell script scenarios. Regardless of type, artifact used as cli command and Unix-like cli interface with arguments, stdin, stdout and stderr file streams. Universal input-output protocol specification for DRCE task is only one limitation for artifact to get the link with algorithm encapsulated inside, data to process and processing results.

Task chain

It is a set of tasks that is united by planning scheduler process, execution environment location and results fetching way. The main differences of tasks that set in chain from single tasks it is that tasks united in chain will be executed sequentially on the same physical host by one hce-node DRCE functional object handler. Tasks that are united in chain have none zero parent Id and set to the DRCE cluster in one request.

Execution environment

The execution environment is represented by the regular HCE cluster with usage of main DRCE functionality. The manager nodes are in balancing replica mode. Main goal of usage this mode it is a parallel competition of data node hosts and handlers of nodes at one host in case of multi-core CPUs. The task need to be fully compatible with execution environment unit configuration, for example, correspondent architecture and executable need to be supported by OS for binary execution and run-time environment like JVM/JRE for Java or PHP, Python, Ruby and so on need to be installed and configured with libraries to be capable to execute tasks code.

Execution environment service

It is a HCE DRCE cluster router that represents the cluster's entry point. The DTM application instance can connect and work with many clusters as well as one cluster can be used by many instances of DTM. Depends on scalability or durability principles the proportion can be one to one, and one to many in recombination. The execution environment service plays the server role and DTM application – the client role. The service represents black-box functional unit with complete automated algorithms and dedicated management tools that is a part of HCE.

Execution environment manager

It is an object that interacts with the execution environment service, makes a DRCE requests and process responses, process requests from the **client interface service** and tasks executor as well as query the task manager to update a tasks state. It works as a broker between tasks executor and execution environment service to make interaction asynchronous.

Planning strategies

This is a set of algorithms that used to assign tasks to the time slots. Those algorithms define principles and conditions for optimizations for execution environment state as well as for tasks execution sequence and so on. Depends on target user's aims the behavior of load-balancing of resources and effectiveness of tasks execution can be changed and tuned up.

Tasks scheduling

It is a process of assigning the tasks to time slots in the schedule. This process acts as planning strategy and can go on demand when new task is got from user client or periodically.

Tasks scheduler or just Scheduler

It is an object that holds the schedule and process assignment of tasks for time slots as well as other basic operations with time slots like movements of past time slots from planned to log, change state and so on. Scheduler processes requests messages from the task manager and **tasks executor** and query the resource manager to update resources load level and amount value.

Data node

It is an HCE DRCE cluster node that is instantiates the DRCE Functional Object. This is down level of cluster hierarchy and hold executor of tasks as well as resources and tasks state notify client.

POSIX thread

This is a regular thread object in terms of target physical host OS platform. It is a low level execution unit for a machine code that is used by OS scheduler to manage CPU and RAM resources. Each CPU kernel executes some code that is represented by one POSIX thread at time. One OS process can to create more than one POSIX thread and try to get more than one CPU kernel to execute parts of own code simultaneously. The number of POSIX threads that process instance (task command from DRCE tasks point of view) can to create is important for tasks planning and scheduling process because reflects the predictable load level and physical resources usage for the process and the task during execution.

Tasks data storage

This is a data storage that used to store the data that was specified with the task and need to be sending to the execution environment. It can be a single-host location regular file system, SQL and NOSQL database, and so on. Depends on tasks specific properties like size of the data, data location it can be single-host and distributed with some scalability principles.

Tasks state update service

It is an object that holds ZMQ protocol based server. It accepts requests for update of tasks state directly from execution environment data nodes and queries the task manager with update information

about tasks state. It works as a broker between task manager and execution environment to make the update tasks state process asynchronous.

Tasks executor

It is an object that implements main tasks execution cycle timing interacts with the scheduler to fetch tasks for time slots and send them to the execution environment manager as well us queries the tasks manager to update tasks state. It works as a main clock processor.

Resource state monitor

It is an object that does monitoring of execution environment resources state and updates resources properties by query the resource manager. It works as an execution environment resources watchdog. Potentially, can be extended with functionality of the execution environment resources manager.

Client interface service

It is an object that accepts client's requests on task and interacts with execution environment manager and tasks manager. Depends on requested operation for task it can be directly forwarded to the execution environment or to the task manager. It works as a broker to make clients requests processing asynchronous.